J. Dubois
I. Elouafiq
R. Xue
S. Yuniyarti
A. Blanc, Ed.
Telecom Bretagne
March 24, 2014

Group 14: c2w protocol specification (Version 2.0)

Abstract

This document provides a sample protocol specification for the binary version of the "c2w" protocol. The c2w protocol is based on the client−server model. The goal of this protocol is to support all the communication requirements of the application "c2w," which allows users to chat while watching a movie.

Table of Contents

1. Introduction

The c2w protocol is an application layer protocol that is intended for the c2w application. The goal of this application is to enable clients to watch videos present in the server and to chat in the same time.

In the c2w application, a user starts by logging in with his/her username. The user must also specify the address and the port number of the server. If the login request is successful, the user joins the main-room where he/she can either chat with other users (currently in the main-room) or choose to join a movie room so he can Chat While Watching a movie. In the main room, the client sees a list of all the users with their availability and a list of all the available movies. In a movie room, a user has access to a list of all the users in the same room. Regardless of the current location of a user (main room or movie room), he/she can select another user to start a private chat. The user can go back to the main room from a movie room and he/she can disconnect by leaving the main room.

Clients can use either UDP or TCP to exchange messages with the server. But once the login request has been sent, the client cannot switch to another Transport layer protocol. A server MAY accept only UDP (respectively TCP) connections.

This document defines the protocol used by the clients to talk with the server and vice-versa. The c2w protocol enables working with an unreliable transport protocol such as UDP.

In our protocol, we have a unique format for all the packets, but since these packets are of different types, there are differences in the data field.

While no security considerations were taken into account, the provided specifications allow improving the c2w protocol for security.

2. Terminology and Abbreviations

This document uses the following terms:

* The capitalized words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [1].

* Id : (or ID) identifier.

* The Id of a movie room and the Id of the corresponding movie have the same meaning.

* UDP : User Datagram Protocol. [2]

* TCP : Transmission Control Protocol. [3]

* AYT : Are You There (see Section 4. Packet Format).

* c2w : Chat While Watching.

* IP : Internet Protocol. [4]

* SRC: Source. Can be either a client or a server.

* ACK: (or Ack) Acknowledgement.

* Message: a text that a certain client needs to send to other client.

3. General Specifications

When a client wants to connect to the server it MUST send a Login Request. The login request does not contain the User Id of the client, since this Id is unknown before establishing connection with the server, but it does contain the username.

The username of the client should be different from other usernames. The server SHOULD make sure that no two clients have the same username (eventhough the User Id can differentiate between different users).

The number of clients MUST be lower than 255. The number of movies MUST be lower than 255.

The number of characters of the user name MUST be lower than 30 characters

The number of characters of the movie name MUST be lower than 100 characters

The length of a message sent by the client MUST be lower than 140 bytes.

Clients MUST communicate with each other through and only through the server.

Whenever the server sends a packet to a certain client the User Identifier MUST be set as the Id of that client.

The acknowledgement (if appropriate) has the same sequence number of the request, and the client MUST wait for the acknowledgement of each request. Note that even the Type Field of the Ack is the same as the Type Field of the request. The sender of a certain request that needs to be acknowledged might not receive an Ack either because the packet was lost or the request was lost. If this is the case and the sender does not receive the acknowledgement in a certain delay of time, the client will re−send the same message with the same sequence number. Moreover, if a packet was sent 3 times and hasn't been acknowledged the client MUST stop sending it to avoid saturating the server.

A client is considered as "connected" if the server receives a Login Request (see next section: Packet format) from that client and sends a acknowledgement back. The server should then proceed by sending the User list and the Movie list to the client. The sequence numbers are synchronized at the beginning. The server has a sequence number for each client.

When a client 'A' needs to send a message to a client 'B', 'A' sends the message to the server (with the Id of 'B' in the destination Id and the Id of 'A' as the User Identifier). The server then forwards the message to 'B' (for the packet sent by the server, the destination's Id will be the Id of 'A' while the User Identifier will be the Id of 'B').

4. Packet format

The following figure (Figure 1) shows the format of a c2w packet. Every packet sent by both the server and the client MUST follow this format.

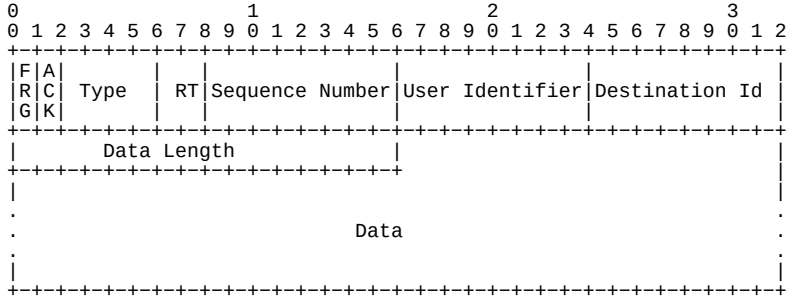


Figure 1

* FRG flag (1 bit): Indicates if the message is the fragmented (refer to Section 10 for more details)

  − 0 if the message is not fragmented OR if it's the last message of a series of messages fragmented.

  − 1 if the message is fragmented, so the client MUST wait for further messages.

* ACK flag (1 bit): Indicates if the packet is an acknowledgement of a received packet

  − 1 if the packet is an acknowledgement.

  − 0 otherwise.

* Type field (4 bits): Indicates the type of the packet

The following table shows the codes for the Type field.

Code (binary)	Corresponding Type
0000	Login Request
1111	Disconnection Request
0001	Message
1000	Room Request
0111	Leave Private Chat Request
0011	Movie List
0101	User List
1100	Message Forward
1010	Private Chat Request
1001	Leave Private Chat Request Forward
0110	AYT
1110	Error Message

Table 1: Binary representation for each type message

- Login Request: Indicates that the corresponding packet is for a login request.

- Disconnection Request: tells the server that the client sending this request wants to disconnect.

Note that leaving the Main Room is considered as a Disconnection Request.

- Message: This type should be set whenever the corresponding packet is a message whether it is a private message, a message meant to be sent to a movie room or a message for the main room.

- Room Request: Set when a client needs to have access to a room (The RT field will then specify which type the room is and the Destination Id will precise the Id of the room).

Note that the request for opening a private chat is considered as a Room Request (having an RT field equal to 10, and a Destination Id equal to the Id of the user to whom the message has to be forwarded). Leaving a Movie Room is considered as sending a request to the Main Room

- Leave Private Chat Request: Indicates that a client wants to leave a private chat with another client.

- Movie List: Indicates that the corresponding packet is a movie list.

- User List: Indicates that the corresponding packet is a user list.

- Message Forward: Forwards a message sent from a client to another client. If the server receives a message (type Message) destined for a single client (i.e: the Room Type field contains a private chat type and the Destination Id field contains the User Identifier of the destination client), it sends a packet of this type for the destination client. If the server receives a message destined for a room, it will send a single message for each client present in that room.

- Private Chat Request: When the server wants to ask a client to open a Private Chat with another client (who requested that Private Chat with the server).

- Leave Private Chat Request Forward: When the server wants to tell a client that another client (with whom he was in a Private Chat) has left the Private Chat.

- AYT (Are You There): With this request the server can verify if a client is still connected. In case this client does not respond back with an AYT Ack, the server will consider that the corresponding client has been disconnected.

- Error Message: Used to communicate an error condition (e.g., malformed packet, username already in use)

Note that the message error is a special type of the acknowledgement, which means that the ACK flag is set to 1 and that it has the same sequence number of the message that caused the error.

If the acknowledgement is an Error Message, the first byte of the data field MUST contain the corresponding error code (see Table 2) .

The following table shows the binary representation of the type errors.

Error Code (binary)	Description
00000001	Username not available
00000010	Id does not exist
00000011	Server has reached its capacity
00000100	Message too long
00000101	Private chat request rejected
00000110	Invalid message

Table 2: Binary representation for each type errors

The type "Invalid message" corresponds to all the other errors.

* The acknowledgement for a message just changes the flag of ACK to 1 by the server. We keep the rest of the message.

* RT (Room Type) field (2 bit): Indicates the type of the corresponding room,

  − 00: Main Room.

  − 01: Movie Room.

  − 10: Private Chat.

  − 11: Not Applicable.

Chat messages and user lists MUST use the appropriate room type, while all other messages, including acknowledgments, MUST use the 'Not Applicable' (11) Room Type.

* Sequence number field (8 bits): Contains the sequence number of the packet being sent,

  − When a connection is established this field is set to 0.

  − This number is increased by 1 if the ACK for the previous packet has been received.

  − When the maximum number is reached (255) the sequence number is reset back to 0.

  − Each client holds his own sequence number with which it interacts with the server.

  − The server holds a different sequence number for each client it interacts with.

* User Identifier field (8 bit): Contains the user identifier of the client. After the communication has been established between a client and the server (i.e., the client sent a Login Request and the server responded with a acknowledgement containing the User Id of that client), when the client and the server communicate with each other the User Id is always equal to the Id of the client. In all chat messages (public or private) this field contains the user ID of the sender of the messages. This way the other clients receiving the message know the author.

  − This number is defined by the server at the connection of the client.

  − Each client has a different unique user identifier.

* Destination Id field (8 bit): The Identifier corresponding to the destination. In case the packet is a:

  − Movie Room User list: This field contains the room ID of the movie room.

  − Message broadcasted to a room: This field contains the corresponding destination room's ID. Note that each Movie Room has a single ID, if this message is sent to the Main Room, this field MUST be set to 0, since the server knows that the message is for the Main Room from the RT (Room Type) field.

  − Private message: This field will contain the corresponding destination user's ID.

  − Otherwise, this field must be set to 0 (including acknowledgments).

* Data Length field (16 bit): Specifies the length of the Data field in bytes. All packets MUST have this field, which can have a value of 0.

* Data field (Variable Length): The data corresponding to each packet. In case the message is a:

− Movie list: the data is represented as in Figure 2. Note that all the fields Name Length and Room Id are 1 byte each.

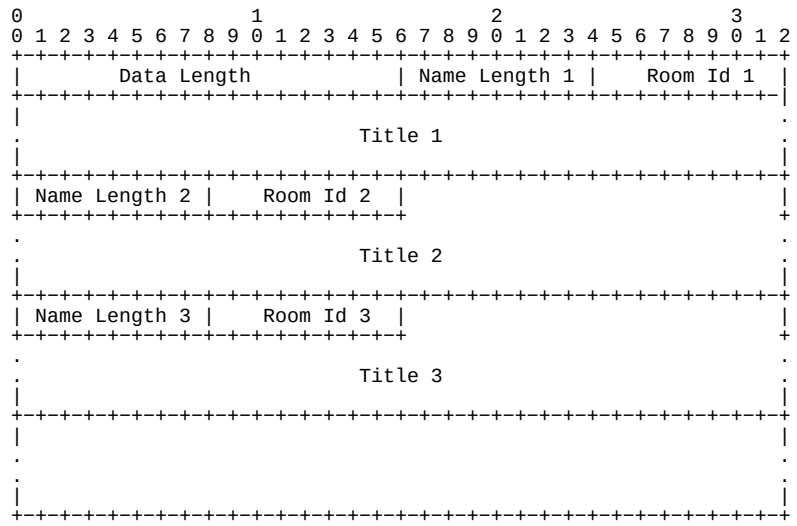


Figure 2

– User list: the data is represented in Figure 3. Note that all the fields Name Length and User Id are 1 byte each. The status field is one bit and the following 7 bits of that byte are reserved and must be set to 0.

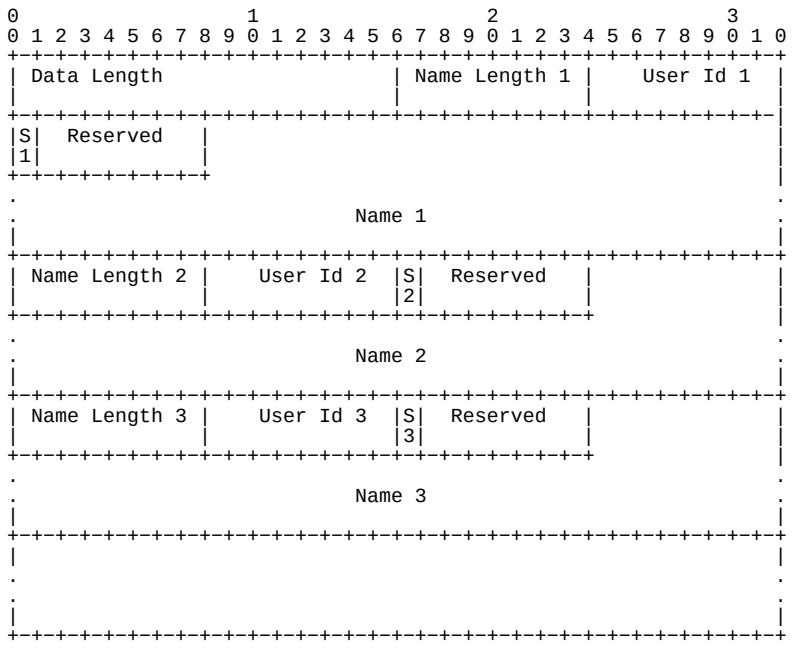


Figure 3

Where S ’i’ : indicates the status of the user number ’i’. If this bit is equal to 1, it means that the user is Available (in the main room). If it is equal to zero, it means that the user is in a movie room. The reserved bits MUST always be set to 0.

– Login Request: The data field will contain the username of the client.

– Message or Message Forward: The data field will contain the message itself.

– Room Request Ack, in case the RT Field is equal to 01 (movie room): The server must send the Port Number and the IP Address. In this

case the the first two bytes of the Data Field contain the port number and the next 4 bytes contain the IP Address. The IP Address must be in the following order: For instance, if the IP Address is 192.0.1.0 the byte following the port number contains 192, the next one will contain 0, then 1, then 0.

5. Sequence Number Selection

Since the c2w protocol is supposed to work on both TCP and UDP, loss and out of order delivery of packets should be taken into account. Hence, to each transmitted packet, a Sequence Number MUST be assigned. The Sequence Number is a number between 0 and 255.

For the Login Request, the Sequence Number is 0. For each client communicating with the server, there are two Sequence Numbers: one held by the client: CSSN (Client Side Sequence Number), the second held by the server: SSSN (Server Side Sequence Number). Note that, the SSSN and CSSN for two different clients are different and that the server holds a different SSSN for each different client. At the beginning, both the SSSN and CSSN are set to 0. Whenever the server sends a packet which is not an acknowledgement, the SSSN related to the corresponding client is incremented by one. If it is the client who sends a packet which is not an acknowledgement the CSSN is incremented by one. (Note that the SSSN and CSSN are equal at the beginning of the connection but can differ afterwards)

Since every single packet is sequenced, each of them can be acknowledged. For each acknowledgement the Sequence Number is set as the acknowledged packet's sequence number.

After each acknowledgement received, the Sequence Number is incremented by one. Once it reaches 255 it should be set back to 0.

Thus, when the server receives a packet from a client the acknowledgement it sends (if appropriate) SHOULD have the same Sequence Number. The same thing goes for the clients.

6. User List Update

In the c2w application clients may enter or leave different rooms frequently. Consequently, the User List held by each user has to be updated regularly.

Every time a client 'A' logs in, logs out or changes his room, the new user list is sent to all the 'interested' clients. Clients are considered to be 'interested' if they are in: the Main Room, the

Movie Room that 'A' got into or the Movie Room that 'A' has just left. Moreover, if a client gets in a Movie Room, he should receive the list of users in that room. In this case, the S 'i' field in the User List packet (see Section 4. Packet Format, − Data Field) is insignificant.

If the client receives different user lists at the same time (for example, a user list update because a new user arrives in the main room and the second list because he rejoins a movie room), he MAY differentiate the list thanks to the Room Type(2 bits in our packet format).

7. Movie Rooms

A movie is associated with each Movie Room. With each movie, a different multicast IP Address is associated. To receive the video flow of a certain movie, clients need to know the IP Address associated with that movie.

To be able to watch a movie, the client MUST send a Room Request to the server where the Room Type is a Movie Room and the Destination Id is the Id of the corresponding movie. When the server approves the request, it sends a acknowledgement containing the IP Address and the port number of that movie (in the data field we will take 6 bytes for the IP address and the port number, the first two bytes for the port number and the other bytes for the IP address) , then it sends the User List of that movie room to the same client and updates the user lists of the other 'interested' (see previous section) clients (See Section 4, Packet Format − Data Field − Room Request Ack).

8. Private Chat

In order for 'A' to open a Private Chat with client 'B', 'B' MUST agree. Hence, to start a private chat, 'A' sends a Room Request to the server where the Room Type is a Private Chat and the Destination Id is that of 'B'. The server will then send a Private Chat Request to 'B' where the Destination Id is that of 'A' (Yes, when 'B' and the server communicate the Destination Id is that of 'A' even if 'A' is the one sending the request), and 'B' should send a Private Chat Ack back to the server. The server will not send an acknowledgement to 'A' until it receives the acknowledgement from 'B'. Once that acknowledgement is received from 'B' the server acknowledges 'A'. If 'B' does not respond to the server, the server MUST send a acknowledgement of type message error to 'A'.

Note that the delay between the request sent by 'A' and the received

acknowledgement might be longer because the server should contact 'B' first. It is RECOMMENDED to take this delay into acount for the timeout delay of the private chat request in the implementation.

9. The AYT request

This request enables the server to verify that a client is still connected.

Note that using the AYT request is OPTIONAL.

10. Fragmentation

When a message exceeds 65507 bytes in length, it MUST be fragmented into packets of at most 65507 bytes. Each single fragment MUST have the first 48 bits as specified in Section 4. Every fragment MUST also contain the data length of the fragment.

The FRG field MUST contain the value 1 for all the fragments except for the last one. The FRG field of the last fragment MUST be set to 0.

The Sequence Number is increased by one for each single fragment. The sender MUST not send a fragment until the acknowledgement of the previous fragment is received. (See Scenario 6:Fragmentation)

Note that whenever a message is not fragmented the FRG field MUST be set equal to 0.

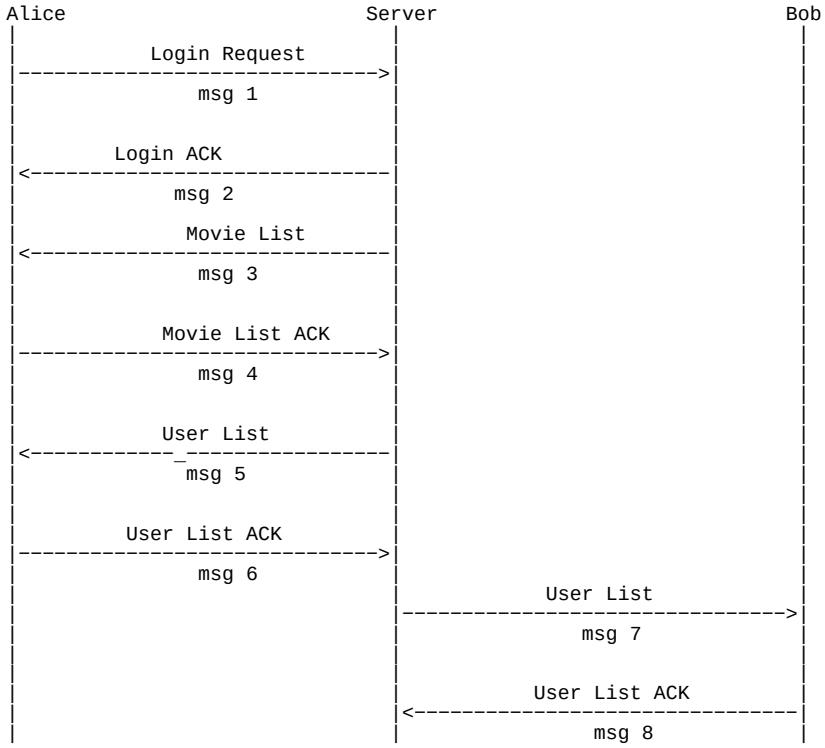
11. Example scenario

All the values of the fields were represented in Hexadecimal notation.

11.1. Scenario 1: Client Login

This first scenario describes the case where a new client 'Alice' (0x61 6c 69 63 65) wants to login. Alice sends a Login Request to the server. Once the server receives that request, it proceeds by sending the Login acknowledgement and User Identifier (for example: 0x0C) to Alice, then the Movie List and the User List. It (the server) will then send the new User List to all of the clients that are in the main room (for example 'Bob' with a User Identifier equal to 0x08).

In this example we assume that there is only one movie in the movie list: 'despicable octopus' (0x64 65 73 70 69 63 61 62 6c 65 20 6f 63 74 6f 70 75 73) and that the only users are Bob (0x42 6f 62) and Alice.



```
-msg 1 - Login Request:
  FRG:              0
  ACK:              0
  Type:            00
  RT:              11
  Sequence Number: 00
  User Id:         00
  Destination Id:  00
  Data Field:
    -> Data Length: 05
    -> Username: 61 6C 69 63 65 (alice)

-msg 2 - Login Ack:
  FRG:              0
  ACK:              1
  Type:            00
  RT:              11
  Sequence Number: 00
  User Id:         0C
  Destination Id : 00

-msg 3 - Movie List
  FRG:              0
  ACK:              0
  Type:            03
  RT:              00
  Sequence Number: 00
  User Id:         0C
  Destination Id:  00
  Data Field:
    -> Data Length: 14
    -> Name Length: 12
    -> Room Id:     08
    -> Movie Name : 64 65 73 70 69 63 61 62 6c 65 20 6f 63 74 6f 70
                    75 73 (Despicable Octopus)

-msg 4 - Movie List Ack:
  FRG:              0
  ACK:              1
  Type:            03
  RT:              11
  Sequence Number: 00
  User Id:         0C
  Destination Id:  00
```



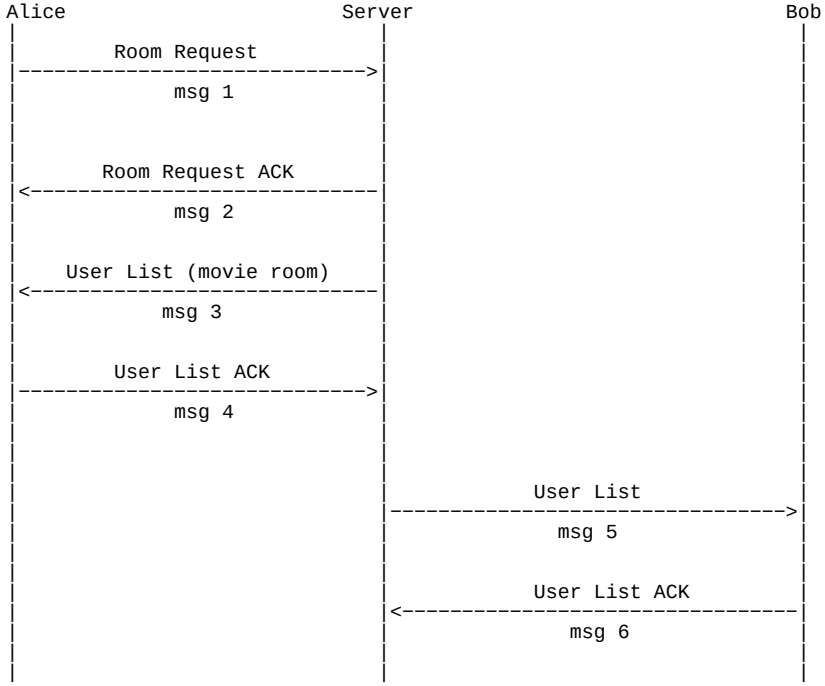
```
-msg 5 - User List:
  FRG:              0
  ACK:              0
  Type:            05
  RT:              00
  Sequence Number: 01
  User Id:         0C
  Destination Id : 00
  Data Field:
    -> Data Length : 0C
    -> Name Length1: 03
    -> Username 1:   42 6F 62 (bob)
    -> User Id 1:    08
    -> Name Length2: 05
    -> Username 2:   61 6C 69 63 65 (alice)
    -> User Id 2:    0C

-msg 6 - User List Ack:
  FRG:              0
  ACK:              1
  Type:            05
  RT:              11
  Sequence Number: 01
  User Id:         0C
  Destination Id:  00

-msg 7 - User List:
  FRG:              0
  ACK:              0
  Type:            05
  RT:              00
  Sequence Number: B4
  User Id:         08
  Destination Id:  00
  Data Field:
    -> Data Length:  0C
    -> Name Length1: 03
    -> Username 1:   42 6F 62 (bob)
    -> User Id 1:    08
    -> Name Length2: 05
    -> Username 2:   61 6C 69 63 65 (alice)
    -> User Id 2:    0C

-msg 8 - User List Ack:
  FRG:              0
  ACK:              1
  Type:            05
  RT:              00
```

```
Sequence Number: B4
User Id:         08
Destination Id:  00
```

11.2. Scenario 2: Client gets into a Movie Room

Alice chooses decides to watch a movi. The server will send the Room Request Acknowledgement (with the Port Number and the IP Address of the movie) and the User List in the related movie room. We assume that Bob is already in that movie room. Consequently, the server will send the User List to Bob for him to update his User List.

We will assume that the requested movie room's Id is equal to 0x08.




```
−msg 1 − Room Request:
FRG:              0
ACK:              0
Type:            08
RT:              01
Sequence Number: 08
User Id:         0C
Destination Id:  08

−msg 2 − Room Request Ack:
FRG:              0
```

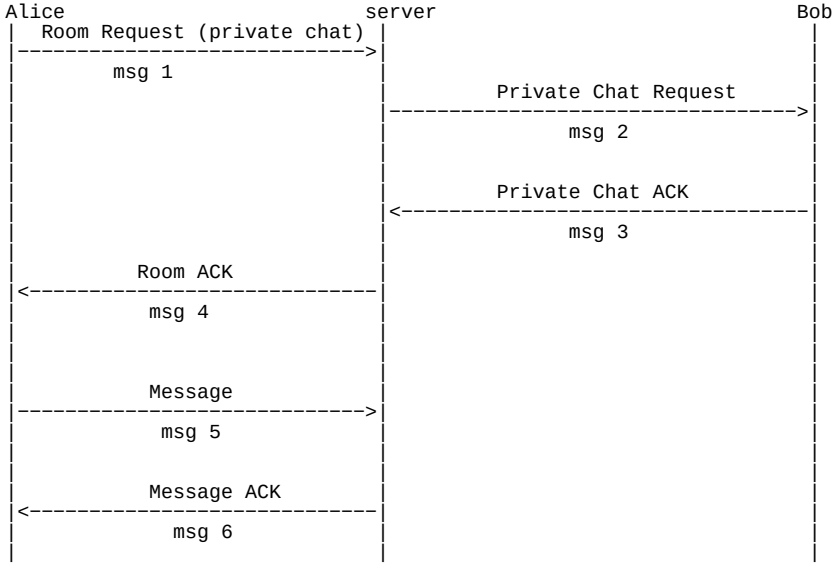
```
ACK:              1
Type:            08
RT:              11
Sequence Number: 08
User Id:         0C
Destination Id:  00
Data Field:
  -> Port Number: 0A
  -> IP Address : 2C 65 2F 00

-msg 3 - User List (movie room):
FRG:              0
ACK:              0
Type:            05
RT:              01
Sequence Number: BB
User Id:         0C
Destination Id   00
Data Field:
  -> Data Length:  0C
  -> Name Length1: 03
  -> Username 1:   42 6F 62 (bob)
  -> User Id 1:    08
  -> Name Length2: 05
  -> Username 2:   61 6C 69 63 65 (alice)
  -> User Id 2:    0C

-msg 4 - User List Ack:
FRG:              0
ACK:              1
Type:            05
RT:              11
Sequence Number: BB
User Id:         0C
Destination Id   00

-msg 5 - User List:
FRG:              0
ACK:              0
Type:            05
RT:              01
Sequence Number: F1
User Id:         08
Destination Id   00
Data Field:
  -> Data Length : 0C
  -> Name Length1: 03
  -> Username 1:   42 6F 62 (bob)
```



```
  -> User Id 1:    08
  -> Name Length2: 05
  -> Username 2:   61 6C 69 63 65 (alice)
  -> User Id 2:    0C

-msg 6 - User List Ack:
FRG:              0
ACK:              1
Type:            05
RT:              11
Sequence Number: F1
User Id:         08
Destination Id:  00
```

11.3. Scenario 3: User begins a private chat

Alice and Bob are in the main room. Alice wants to begin a private chat with Bob. The User Id of Alice is 12(0x0c),and the User Id of Bob is 8 (0x08).



− msg 1 − Room Request (Private Chat):

FRG : 00

ACK : 00

Type : 08

RT : 02

Sequence Number: 2F

User Id : 0C

Destination Id : 08

− msg 2 − Private Chat Request:

FRG : 00

ACK : 00

Type : 0A

RT : 02

Sequence Number: AB

User Id : 08

Destination Id : 0C

− msg 3 − Private Chat Ack:

FRG : 00

ACK : 01

Type : 0A

RT : 02

Sequence Number: AB

User Id : 08

Destination Id : 0C

− msg 4 − Room Ack:

FRG : 00

ACK : 01

Type : 08

RT : 02

Sequence Number: 2F

User Id : 0C

Destination Id : 08

− msg 5 − Message:

FRG : 00

ACK : 00

Type : 01

RT : 02

Sequence Number: 30

User Id : 0C

Destination Id : 08

Data Field :

−> Data Length: 05

−> Message : 65 6c 6c 6f (hello)

− msg 6 − Message Ack:

FRG : 00

ACK : 01

Type : 01

RT : 02

Sequence Number: 30

User Id : 0C

Destination Id : 08

− msg 7 − Message Forward:

FRG : 00

ACK : 00

Type : 01

RT : 02

Sequence Number: AC

User Id : 08

Destination Id : 0C

Data Field :

−> Data Length: 05

−> Message : 65 6c 6c 6f (hello)

− msg 8 − Message Received Ack:

FRG : 00

ACK : 01

Type : 0C

RT : 02

Sequence Number: AC

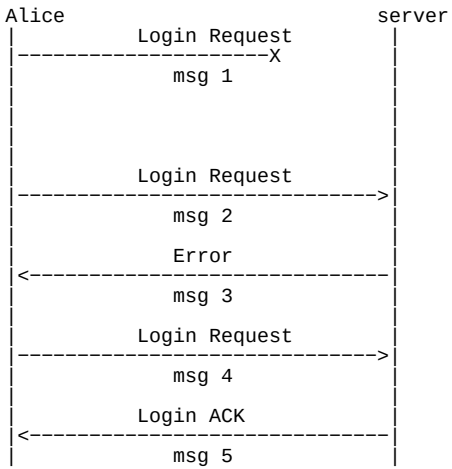
User Id : 08

Destination Id : 0C

11.4. Scenario 4: Packet loss and errors

Another user named Alice wants to login. Alice enters ’alice’ as a username. The first Login Request sent by Alice will be lost. Consequently, after the timeout the client will send the same Login Request again (with the same Sequence Number). Since there is another client with the same username (’alice’) When the server receives the Login Request it will send an Error of Error Type 0x01

to the client. Then the client will change the username to 'alicewonderland' and send a new Login Request that will be acknowledged by the server.



- msg 1 - Login Request:

FRG : 00

ACK : 00

Type : 00

RT : 00

Sequence Number: 00

User Id : 00

Destination Id : 00

Data Field:

-> Data Length: 05

-> Username : 61 6C 69 63 65 (alice)

- msg 2 - Login Request:

FRG : 00

ACK : 00

Type : 00

RT : 00

Sequence Number: 00

User Id : 00

Destination Id : 00

Data Field:

-> Data Length: 05

-> Username : 61 6C 69 63 65 (alice)

- msg 3 - Error:

FRG : 00

ACK : 01

Type : 0e

RT : 00

Sequence Number: 00

User Id : 00

Destination Id : 00

Data Field:

-> Data Length: 01

-> Error Type : 01

- msg 4 - Login Request:

FRG : 00

ACK : 00

Type : 00

RT : 00

Sequence Number: 00

User Id : 00

Destination Id : 00

Data Field:

-> Data Length: 15

-> Username : 0e 61 6c 69 63 65 57 6f 6e 64 65 72 6c 61 6e 64
(alicewonderland)

- msg 5 - Login Ack:

FRG : 00

ACK : 01

Type : 00

RT : 00

Sequence Number: 00

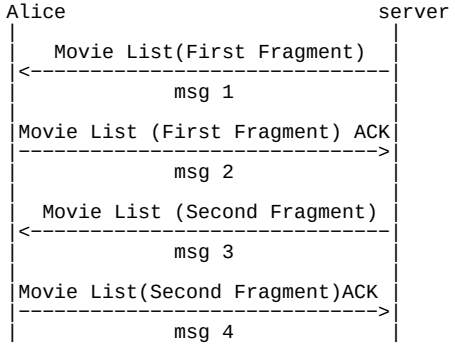
User Id : 0C

Destination Id : 00

11.5. Scenario 5: Fragmentation

Alice receives a Movie List from the server. This time, the packet of the movie list is longer than 64kbits. Consequently, the packet is going to be fragmented. The FRG field of the first fragment will be equal to 1, while that of the second fragment (which is the last one in this case) is going to be equal to 0.

Thus, when the client (Alice) receives the First Fragment of the Movie List she acknowledges this fragment and waits for the next fragments to arrive. Once the packet of an FRG field equal to 0 arrives the client knows that it is the last fragment.



- msg 1 - Movie List (First Fragment):

FRG : 01

ACK : 00

Type : 03

RT : 00

Sequence Number: 00

User Id : 0C

Destination Id : 00

Data Field:

-> Data Length : 200

-> Name Length1: 12

-> Room Id 1 : 08

-> Movie Name 1: 64 65 73 70 69 63 61 62 6c 65 20 6f 63 74 6f 70 75 73 (despicable octopus)

...

...

...

-> Name Length80: 1E

-> Room Id 80 : 50

-> Movie Name 80: 61 6c 6c 20 79 6f 75 72 20 62 61 73 65 20 61 72 65 20 62 65 6c 6f 6e 67 20 74 6f 20 75 73 (all your base are belong to us)

- msg 2 - Movie List (First Fragment) Ack:

FRG : 00

ACK : 01

Type : 03

RT : 00

Sequence Number: 00

User Id : 0C

Destination Id : 00

- msg 3: Movie List (Second Fragment)

FRG : 00

ACK : 00

Type : 03

RT : 00

Sequence Number: 01

User Id : 0C

Destination Id : 00

Data Field:

-> Name Length: 24

-> Room Id : 51

-> Movie Name : 66 72 65 65 7a 69 6e 67 20 77 6f 6f 6b 69 65 20 61 6e 64 20 74 68 65 20 73 65 63 72 65 74 20 6e 69 6e 6a 61 (freezing wookie and the secret ninja)

- msg 4: Movie List (Second Fragment) Ack

FRG : 00

ACK : 01

Type : 03

RT : 00

Sequence Number: 01

User Id : 0C

Destination Id : 00

12. Conclusion

In our proposition, we gave all the packets a common format whith different options available within different packet types. Each packet is sequenced and acknowledged to insure the reliability of the protocol. The example scenarios in the end illustrate how the protocol satisfies the different functions of the c2w application.

However, the format of the packets was defined in such a way that some bits may remain insignificant depending on the type of the packet. There are more issues to be explored. For instance, some specifications such as the Update of user lists at each change can cause overflows, packet loss and errors. So far, we have avoided security issues but these can be improved.

13. Normative references

[1] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[3] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[4] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

Authors' Addresses

Jean-Baptiste Dubois
Telecom Bretagne
Brest,
France

Ismail Elouafiq
Telecom Bretagne
Brest,
France

Rulin Xue
Telecom Bretagne
Brest,
France

Sisihlia Yuniyarti
Telecom Bretagne
Brest,
France

Alberto Blanc (editor)
Telecom Bretagne
Rennes
France